

Authenticate Using Yahoo with Java Script

You can let your users authenticate with Firebase using OAuth providers like Yahoo by integrating generic OAuth Login into your app using the Firebase SDK to carry out the end to end sign-in flow.

Before you begin

To sign in users using Yahoo accounts, you must first enable Yahoo as a sign-in provider for your Firebase project:

1. [Add Firebase to your JavaScript project \(/docs/web/setup\)](#).
2. In the Firebase console, go to **Security** > **Authentication** (https://console.firebase.google.com/project/_/authentication/).
3. In the **Sign-in method** tab, enable the **Yahoo** sign-in provider.
4. Add the **Client ID** and **Client Secret** from that provider's developer console to the provider configuration:
 - a. To register a Yahoo OAuth client, follow the Yahoo developer documentation on [registering a web application with Yahoo](#) (https://developer.yahoo.com/oauth2/guide/openid_connect/getting_started.html).
Be sure to select the two OpenID Connect API permissions: **profile** and **email**.
 - b. When registering apps with these providers, be sure to register the `*.firebaseapp.com` domain for your project as the redirect domain for your app.
5. Click **Save**.

Handle the sign-in flow with the Firebase SDK

If you are building a web app, the easiest way to authenticate your users with Firebase using their Yahoo accounts is to handle the entire sign-in flow with the Firebase JavaScript SDK.

To handle the sign-in flow with the Firebase JavaScript SDK, follow these steps:

1. Create an instance of an **OAuthProvider** using the provider ID **yahoo.com**.

Web modular API	Web namespaced API
<pre>import { OAuthProvider } from "firebase/auth"; const provider = new OAuthProvider('yahoo.com'); :92252bd558cfe05/snippets/auth-next/yahoo-oauth/auth_yahoo_provider_create.js#L8-L10)</pre>	

2. **Optional:** Specify additional custom OAuth parameters that you want to send with the OAuth request.

Web modular API	Web namespaced API
<pre>provider.setCustomParameters({ // Prompt user to re-authenticate to Yahoo. prompt: 'login', // Localize to French. language: 'fr' }); 92252bd558cfe05/snippets/auth-next/yahoo-oauth/auth_yahoo_provider_params.js#L8-L13)</pre>	

For the parameters Yahoo supports, see the [Yahoo OAuth documentation](https://developer.yahoo.com/oauth2/guide/openid_connect/getting_started.html) (https://developer.yahoo.com/oauth2/guide/openid_connect/getting_started.html). Note that you can't pass Firebase-required parameters with `setCustomParameters()`. These parameters are **client_id**, **redirect_uri**, **response_type**, **scope** and **state**.

3. **Optional:** Specify additional OAuth 2.0 scopes beyond **profile** and **email** that you want to request from the authentication provider. If your application requires access to private user data from Yahoo APIs, you'll need to request permissions to Yahoo APIs under **API Permissions** in the Yahoo developer console. Requested OAuth scopes must be exact matches to the preconfigured ones in the app's API permissions. For example if, read/write access is requested to user contacts and preconfigured in the app's API

permissions, `sdct-w` has to be passed instead of the readonly OAuth scope `sdct-r`. Otherwise, the flow will fail and an error would be shown to the end user.

<u>Web</u> modular API	<u>Web</u> namespaced API
<pre>// Request access to Yahoo Mail API. provider.addScope('mail-r'); // Request read/write access to user contacts. // This must be preconfigured in the app's API permissions. provider.addScope('sdct-w'); 92252bd558cfe05/snippets/auth-next/yahoo-oauth/auth_yahoo_provider_scopes.js#L8-L12</pre>	

To learn more, refer to the [Yahoo scopes documentation](https://developer.yahoo.com/oauth2/guide/yahoo_scopes/) (https://developer.yahoo.com/oauth2/guide/yahoo_scopes/).

4. Authenticate with Firebase using the OAuth provider object. You can prompt your users to sign in with their Yahoo Accounts either by opening a pop-up window or by redirecting to the sign-in page. The redirect method is preferred on mobile devices.

- To sign in with a pop-up window, call `signInWithPopup`:

<u>Web</u> modular API	<u>Web</u> namespaced API
<pre>import { getAuth, signInWithPopup, OAuthProvider } from "firebase const auth = getAuth(); signInWithPopup(auth, provider) .then((result) => { // IdP data available in result.additionalUserInfo.profile // ... // Yahoo OAuth access token and ID token can be retrieved by const credential = OAuthProvider.credentialFromResult(result) const accessToken = credential.accessToken; const idToken = credential.idToken; }) .catch((error) => { // Handle error. });</pre>	

52bd558cfe05/snippets/auth-next/yahoo-oauth/auth_yahoo_signin_popup.js#L8-L23)

- To sign in by redirecting to the sign-in page, call **signInWithRedirect**:

Follow the [best practices](#) (/docs/auth/web/redirect-best-practices) when using **signInWithRedirect**, **linkWithRedirect**, or **reauthenticateWithRedirect**.

```
firebase.auth().signInWithRedirect(provider);
```

After the user completes sign-in and returns to the page, you can obtain the sign-in result by calling **getRedirectResult**.

<u>Web</u> modular API	<u>Web</u> namespaced API
<pre>import { getAuth, signInWithRedirect } from "firebase/auth"; const auth = getAuth(); signInWithRedirect(auth, provider); </pre> 5c92252bd558cfe05/snippets/auth-next/yahoo-oauth/auth_yahoo_signin_redirect.js#L8-L11)	

On successful completion, the OAuth ID token and access token associated with the provider can be retrieved from the **firebase.auth.UserCredential** object returned.

Using the OAuth access token, you can call the [Yahoo API](#) (<https://developer.yahoo.com/oauth2/guide/apirequests/>).

For example, to get the basic profile information, the following REST API can be called:

```
curl -i -H "Authorization: Bearer ACCESS_TOKEN" https://social.yahooapis.com
```

Where **YAHOO_USER_UID** is the Yahoo user's ID which can be retrieved from the **firebase.auth().currentUser.providerData[0].uid** field or from **result.additionalUserInfo.profile**.

5. While the above examples focus on sign-in flows, you also have the ability to link a Yahoo provider to an existing user using `linkWithPopup/linkWithRedirect`. For example, you can link multiple providers to the same user allowing them to sign in with either.

<u>Web</u> modular API	<u>Web</u> namespaced API
<pre> import { getAuth, linkWithPopup, OAuthProvider } from "firebase/auth"; const provider = new OAuthProvider('yahoo.com'); const auth = getAuth(); linkWithPopup(auth.currentUser, provider) .then((result) => { // Yahoo credential is linked to the current user. // IdP data available in result.additionalUserInfo.profile. // Get the OAuth access token and ID Token const credential = OAuthProvider.credentialFromResult(result); const accessToken = credential.accessToken; const idToken = credential.idToken; }) .catch((error) => { // Handle error. }); </pre>	

1190c92252bd558cfe05/snippets/auth-next/yahoo-oauth/auth_yahoo_link_popup.js#L8-L24)

6. The same pattern can be used with `reauthenticateWithPopup/reauthenticateWithRedirect` which can be used to retrieve fresh credentials for sensitive operations that require recent login.

<u>Web</u> modular API	<u>Web</u> namespaced API
<pre> import { getAuth, reauthenticateWithPopup, OAuthProvider } from "firebase/auth"; const provider = new OAuthProvider('yahoo.com'); const auth = getAuth(); reauthenticateWithPopup(auth.currentUser, provider) .then((result) => { // User is re-authenticated with fresh tokens minted and </pre>	

```

// should be able to perform sensitive operations like account
// deletion and email or password update.
// IdP data available in result.additionalUserInfo.profile.

// Get the OAuth access token and ID Token
const credential = OAuthProvider.credentialFromResult(result);
const accessToken = credential.accessToken;
const idToken = credential.idToken;
})
.catch((error) => {
  // Handle error.
});

```

3c92252bd558cfe05/snippets/auth-next/yahoo-oauth/auth_yahoo_reauth_popup.js#L8-L26)

+ Handling account-exists-with-different-credential Errors

If you enabled the **One account per email address** setting in the [Firebase console](https://console.firebase.google.com/) (<https://console.firebase.google.com/>), when a user tries to sign in to a provider (such as Yahoo) with an email that already exists for another Firebase user's provider (such as Google), the error `auth/account-exists-with-different-credential` is thrown along with an `AuthCredential` object (Yahoo credential). To complete the sign in to the intended provider, the user has to sign first to the existing provider (Google) and then link to the former `AuthCredential` (Yahoo credential).

Popup mode

If you use `signInWithPopup`, you can handle `auth/account-exists-with-different-credential` errors with code like the following example:

```

import {
  getAuth,
  linkWithCredential,
  signInWithPopup,
  OAuthProvider,
} from "firebase/auth";

```

```
try {
  // Step 1: User tries to sign in using Yahoo.
  let result = await signInWithPopup(getAuth(), new OAuthProvider());
} catch (error) {
  // Step 2: User's email already exists.
  if (error.code === "auth/account-exists-with-different-credential") {
    // The pending Yahoo credential.
    let pendingCred = error.credential;

    // Step 3: Save the pending credential in temporary storage,

    // Step 4: Let the user know that they already have an account
    // but with a different provider, and let them choose another
    // sign-in method.
  }
}

// ...

try {
  // Step 5: Sign the user in using their chosen method.
  let result = await signInWithPopup(getAuth(), userSelectedProvider);

  // Step 6: Link to the Yahoo credential.
  // TODO: implement `retrievePendingCred` for your app.
  let pendingCred = retrievePendingCred();

  if (pendingCred !== null) {
    // As you have access to the pending credential, you can directly call the
    // link method.
    let user = await linkWithCredential(result.user, pendingCred);
  }

  // Step 7: Continue to app.
} catch (error) {
  // ...
}
```

Redirect mode

This error is handled in a similar way in the redirect mode, with the difference that the pending credential has to be cached between page redirects (for example, using session storage).

⊕ Advanced: Handle the sign-in flow manually

Unlike other OAuth providers supported by Firebase such as Google, Facebook, and Twitter, where sign-in can directly be achieved with OAuth access token based credentials, Firebase Auth does not support the same capability for providers such as Yahoo due to the inability of the Firebase Auth server to verify the audience of Yahoo OAuth access tokens. This is a critical security requirement and could expose applications and websites to replay attacks where a Yahoo OAuth access token obtained for one project (attacker) can be used to sign in to another project (victim). Instead, Firebase Auth offers the ability to handle the entire OAuth flow and the authorization code exchange using the OAuth client ID and secret configured in the Firebase Console. As the authorization code can only be used in conjunction with a specific client ID/secret, an authorization code obtained for one project cannot be used with another.

If these providers are required to be used in unsupported environments, a third party OAuth library and [Firebase custom authentication](#)

(<https://firebase.google.com/docs/auth/admin/create-custom-tokens>) would need to be used. The former is needed to authenticate with the provider and the latter to exchange the provider's credential for a custom token.

Authenticate with Firebase in a Chrome extension

If you are building a Chrome extension app, see the [Offscreen Documents guide](#) ([/docs/auth/web/chrome-extension#use_offscreen_documents](/docs/auth/web/chrome-extension#use_offscreen_documents)).

+ Customizing the redirect domain for Yahoo sign-in

On project creation, Firebase will provision a unique subdomain for your project: `https://my-app-12345.firebaseio.com`.

This will also be used as the redirect mechanism for OAuth sign in. That domain would need to be allowed for all supported OAuth providers. However, this means that users may see that domain while signing in to Yahoo before redirecting back to the application: **Continue to: `https://my-app-12345.firebaseio.com`**.

To avoid displaying your subdomain, you can set up a custom domain with Firebase Hosting:

1. Follow steps 1 through 3 in [Set up your domain for Hosting \(/docs/hosting/custom-domain\)](/docs/hosting/custom-domain). When you verify your domain ownership, Hosting provisions an SSL certificate for your custom domain.
2. Add your custom domain to the list of authorized domains in the [Firebase console](https://console.firebase.google.com/) (`https://console.firebase.google.com/`): `auth.custom.domain.com`.
3. In the Yahoo developer console or OAuth setup page, whitelist the URL of the redirect page, which will be accessible on your custom domain:
`https://auth.custom.domain.com/__/auth/handler`.
4. When you initialize the JavaScript library, specify your custom domain with the `authDomain` field:

```
var config = {
  apiKey: '...',
  // Changed from 'PROJECT_ID.firebaseio.com'.
  authDomain: 'auth.custom.domain.com',
  databaseURL: 'https://PROJECT_ID.firebaseio.com',
  projectId: 'PROJECT_ID',
  storageBucket: 'PROJECT_ID.firebaseio.com',
  messagingSenderId: 'SENDER_ID'
};
firebase.initializeApp(config);
```

Next steps

After a user signs in for the first time, a new user account is created and linked to the credentials—that is, the user name and password, phone number, or auth provider information—the user signed in with. This new account is stored as part of your Firebase project, and can be used to identify a user across every app in your project, regardless of how the user signs in.

- In your apps, the recommended way to know the auth status of your user is to set an observer on the `Auth` object. You can then get the user's basic profile information from the `User` object. See [Manage Users](/docs/auth/web/manage-users) (/docs/auth/web/manage-users).
- In your Firebase Realtime Database and Cloud Storage [Security Rules](/docs/database/security/user-security) (/docs/database/security/user-security), you can get the signed-in user's unique user ID from the `auth` variable, and use it to control what data a user can access.

You can allow users to sign in to your app using multiple authentication providers by [linking auth provider credentials to an existing user account](/docs/auth/web/account-linking). (/docs/auth/web/account-linking)

To sign out a user, call `signOut` (/docs/reference/js/auth#signout):

Web modular API	Web namespaced API
<pre>import { getAuth, signOut } from "firebase/auth"; const auth = getAuth(); signOut(auth).then(() => { // Sign-out successful. }).catch((error) => { // An error happened. });</pre>	

c4c841681cf3642d5ba41190c92252bd558cfe05/snippets/auth-next/index/auth_sign_out.js#L8-L15)

Except as otherwise noted, the content of this page is licensed under the [Creative Commons Attribution 4.0 License](https://creativecommons.org/licenses/by/4.0/) (https://creativecommons.org/licenses/by/4.0/), and code samples are licensed under the [Apache 2.0 License](https://www.apache.org/licenses/LICENSE-2.0) (https://www.apache.org/licenses/LICENSE-2.0). For details, see the [Google Developers Site Policies](https://developers.google.com/site-policies) (https://developers.google.com/site-policies). Java is a registered trademark of Oracle and/or its affiliates.

Last updated 2026-07-10 UTC.

